ICARO

Nuestro compromiso es con el *futuro*.

Back End

Clase 6

ICARO

¿Qué veremos hoy?

Hoy vamos a continuar con el módulo de **Back-End**, donde veremos lo que son los **middlewares**, validadores que podemos implementar haciendo uso de express, como implementar esto último mediante el uso de **middlewares** y finalmente el concepto de paquete npm morgan.

Middlewares

Middleware son funciones que tienen acceso al objeto de solicitud (req), al objeto de respuesta (res) y a la siguiente función de middleware en el ciclo de solicitud/respuestas de la aplicación. Es posible, a su vez, tener encadenamientos del mismo, invocando al siguiente ejecutando la funcion next.

Las funciones de middleware pueden realizar las siguientes tareas:

- Ejecutar cualquier código.
- Realizar cambios en la solicitud y los objetos de respuesta.

Ejemplos Middlewares

```
var app = express();

app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});
```

Este ejemplo muestra una función de middleware sin ninguna vía de acceso de montaje. La función se ejecuta cada vez que la aplicación recibe una solicitud.
Algo muy importante a tener en cuenta, un middleware puede ser utilizado para manejo de errores, mostramos cómo se implementa esto a continuación:

Ejemplos Middlewares. Manejo de errores

```
app.use(function(err, req, res, next) {
  console.error(err.stack);
  res.status(500).send('Something broke!');
});
```

El middleware de manejo de errores se define al final, después de otras llamadas de rutas y app.use(); por ejemplo:

Ejemplos Middlewares. Manejo de errores

```
var bodyParser = require('body-parser');
var methodOverride = require('method-override');

app.use(bodyParser());
app.use(methodOverride());
app.use(function(err, req, res, next) {
  // logic
});
```

Las respuestas desde una función de middleware pueden estar en el formato que prefiera, por ejemplo, una página de errores HTML, un mensaje simple o una serie JSON.

Validaciones. Express Validator

A fin de introducir el concepto, recordemos el formulario que se implementó la última clase, donde se ingresaba nombre y apellido. En este punto, como podremos asegurar que por ejemplo, el nombre tenga una longitud de al menos 6 caracteres?

La manera de implementar dicha validación, es a traves del **express-validator**. El mismo de instala ejecutando:

npm install express-validator

Veamos a nivel implementación/código como sería el resultado final

Validaciones. Express Validator

```
app.post('/form', [
  check('name').isLength({ min: 3 }),
  check('email').isEmail(),
  check('age').isNumeric()
], (req, res) => {
  const errors = validationResult(req)
  if (!errors.isEmpty()) {
    return res.status(422).json({ errors: errors.array() })
  }
const name  = req.body.name
const email = req.body.email
const age   = req.body.age
})
```

Cada **check()** call acepta el nombre del parámetro como argumento. Entonces llamamos al metodo **'validationResult()'** para verificar que no hubo errores de validación.

Validaciones. Express Validator

El módulo de **express-validator** maneja un tipo de mensaje predeterminado:

```
{
  "errors": [{
    "location": "body",
    "msg": "Invalid value",
    "param": "email"
  }]
}
```

En este caso, si el correo es invalido, se mostraria ese formato de error. Dicho mensaje puede ser sobreescrito, usando el metodo **'withMessage'**. A continuación mostramos un ejemplo

Express Validator. Manejo de errores

```
check('name')
  .isAlpha()
  .withMessage('Must be only alphabetical chars')
  .isLength({ min: 10 })
  .withMessage('Must be at least 10 chars long')
```

En este ejemplo, si no se recibe un valor válido en el campo nombre del formulario de longitud minima 10, se mostrará el mensaje de error “Debe ser de al menos 10 caracteres de largo, o en su defecto, Debe ser solo caracteres alfabéticos.

Ejemplo integrador. Express. Validadores

Continuando con nuestro formulario, de la clase pasada, teníamos el siguiente escenario:

Ejemplo integrador. Express. Validadores

Si ingresamos en el campo nombre, un valor de longitud menor a 5, obtendremos:

Ejemplo integrador. Express. Validadores

Ese resultado se logra gracias a que:

1) Se instalo express-validator
2) Se declara que funciones de dicho modulo usaremos:

```
const { body, validationResult } = require('express-validator');
```

3) Dentro de nuestro controlador, agregamos los siguientes cambios:

```
router.post("/",
body('nombre').isLength({ min: 5 }),
(req, res) => {
    const errors = validationResult(req)
    ProcesarFormulario(req, res, errors)
});
```

Ejemplo integrador. Express. Validadores

De lo anterior notamos :

1) El uso de la función que denominamos body, y de la validationResult. De esas dos invocaciones obtenemos un array que contendrá todos los errores de validación. En este caso, solo se aplica al nombre, pero podríamos extenderlo a otros campos también.

```
router.post("/",
body('nombre').isLength({ min: 5 }),
(req, res) => {
    const errors = validationResult(req)
    ProcesarFormulario(req, res, errors)
});
```

Paquete Npm Morgan

Morgan es un Middleware de nivel de solicitud HTTP. Es una gran herramienta que registra las requests junto con alguna otra información dependiendo de su configuración y el valor predeterminado utilizado. Demuestra ser muy útil durante la depuración y también si desea crear archivos de registro

¡Vamos al código!

No olvidemos las clases de consulta!

ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar

Muchas gracias!

ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar